JS Assignment

Topics intended to be covered in this assignment

* Function Expressions, Arrow Functions, Pure and Impure functions
* Operators and Date functions
* Destructuring and Spread operation

Questions

1. **What is the difference in hoisting behaviour between a function created using function declaration syntax and that created using a function expression syntax? Give an example.**

Ans.

Function Declaration is hoisted, meaning it can be called before it is defined in code. Function Expression is not hoisted, or is only partially hoisted (variable is hoisted, not the function definition).

//function declaration

sayHi(); // It will be work

function sayHi() {

console.log("Hi!");

}

//function expression

sayHello(); // It will not work

var sayHello = function () {

console.log("Hello!");

};

1. **Write the usage of following functions / operators and give an example.**
   1. **Date.now()**

Ans.

Date.now(): It will return the current timestamp in milliseconds since Jan 1, 1970.

console.log(Date.now()); // Output - 1755148820174

* 1. **?? (nullish coalescing operator)**

Ans.

?? (Nullish Coalescing Operator): It returns the right-hand operand if the left-hand operand is null or undefined.

let name = null;

let displayName = name ?? "Guest";

console.log(displayName); // "Guest"

* 1. **?. (optional chaining operator)**

Ans.

?. (Optional Chaining Operator): Safely accesses nested object properties without throwing errors if an intermediate property is null or undefined.

const user = { profile: null };

console.log(user.profile?.name); // undefined

1. **Write 2 JS statements, where:**
   1. **1st one will split the words of the sentence I love going walkies into an array.**

**Ans.**

let sentence = "I love going walkies";

let words = sentence.split(" ");

* 1. **2nd one will assign the elements of the resultant array into 4 variables (in a single statement).**

**Ans.**

let [w1, w2, w3, w4] = words;

**<< Given object for Questions 4, 5 (and possibly 6)>>**

**const myObject = {**

**x1: "Samba",**

**x2: {**

**x3: {**

**x4: {},**

**x5: "Rails",**

**},**

**x6: {**

**x7: -1,**

**x8: [25, 8, 4, 10]**

**},**

**}**

**};**

1. **For the given object, write a single line destructuring assignment to get these values:**
   1. **x3 (store result in variable name y)**
   2. **2nd element of x8 (use any variable name for result)**

**Ans.**

const { x2: { x3: y, x6: { x8: [, secondElement] } } } = myObject;

1. **Write JS code to create another object (named newObject), which contains all attributes from the given objects, and the following key value pairs:**
   1. **Key: x20  
      Value: “Shinko”**
   2. **Key: x21  
      Value: [5, 40, 73, 19]**

**Ans.**

const newObject = { ...myObject, x20: "Shinko", x21: [5, 40, 73, 19] };

1. **Create a new array newArray , which combines elements from arrays stored inside attributes x8 and x21 of the object created in Q5.**

**Ans.**

const newArray = [...newObject.x2.x6.x8, ...newObject.x21];

1. **For the given JS program:**

**let aRandomNum = 37;**

**function f1(num) {**

**console.log(num \* 3);**

**}**

**function f2() {**

**aRandomNum \*= 3;**

**console.log(aRandomNum);**

**}**

**f1(aRandomNum);**

**f2();**

**f1(aRandomNum);**

* 1. **Out of the 2 functions f1 and f2, which one is pure and which one is impure? Give reasons for the conclusion.**

Ans.

Here, f1 is the pure function and f2 is the impure function

Reason:

f1 is pure: Its output depends only on its input and has no side effects.

f2 is impure: It modifies external state (aRandomNum).

* 1. **What will be the output of the program (the 3 output values)?**

**Ans.**

let aRandomNum = 37;

function f1(num) {

console.log(num \* 3);

}

function f2() {

aRandomNum \*= 3;

console.log(aRandomNum);

}

f1(aRandomNum); // Output : 111

f2(); // Output : 111

f1(aRandomNum); // Output : 333

1. **In case of arrow functions, in which case the parenthesis for function parameters can be omitted? Give an example.**

Ans. Parentheses can be omitted when the arrow function has exactly one parameter.

const square = x => x \* x;

console.log(square(4)); // 16

1. **Can arrow functions be used as object methods? Why or why not?**

Ans.

No, arrow functions should not be used as object methods if you rely on this, because arrow functions do not have their own this — they inherit it from the surrounding scope.

const obj = {

value: 10,

method: () => {

console.log(this.value); // `this` is not `obj`

}

};

obj.method(); // undefined

Use regular function expressions for object methods that use this.